Intel Atom Base teaching pendant optimized for industrial robots.

DTP7H-P
User's API Guide manual

(R1) Version

DAINCUBE Corp.
Intel Atom Base Windows System

FORM 170703F – 2017.07.03

DTP7H-P User's API Guide manual
Form 170703F-170703— July, 2017

DAINCUBE Corp.
Web: www.daincube.com
E-mail: sales@daincube.com
Tel: 82-32-329-9783~4
Fax: 82-32-329-9785

#401-701, Bucheon TechnoPark 4-Danji,
655 Pyeongcheon-ro, Wonmi-gu, Bucheon-Si,
Gyeonggi-Do, Republic of Korea

Preface

Important information
This documentation is intended for qualified audience only. The product described herein is not an end user product. It was developed and manufactured for further processing by trained personnel.

Disclaimer
Although this document has been generated with the utmost care no warranty or liability for correctness or suitability for any particular purpose is implied. The information in this document is provided “as is” and is subject to change without notice.

Trademarks
All used product names, logos or trademarks are property of their respective owners.

Product support
DAINCUBE Corp.
Web: www.daincube.com
E - MAIL: sales@daincube.com

Safety precautions

Be sure to observe all of the following safety precautions.

Strict observance of these warning and caution indications are a MUST for preventing accidents, which could result in bodily injury and substantial property damage. Make sure you fully understand all definitions of these terms and related symbols given below, before you proceed to the manual.

Symbols

The following symbols may be used in this specification:

Warning

Warnings indicate conditions that, if not observed, can cause personal injury.

Caution

Cautions warn the user about how to prevent damage to hardware or loss of data.

Note

Notes call attention to important information that should be observed.

Revision history

Revision	Data	Comment
Version 1.0	2018.03.26	Initial version
Version 1.0.1	2019.04.18	Touch, Keypad Enable/Disable Function add
Version 1.0.2	2019.05.28	Switch Function remove

Contents

1. Introduction

This document describes the DTP7H-D Key,LED and Buzzer to help users develop applications more easily. LED and Buzzer are controlled by serial communication or controlled by keyboard event. Daincube provides all device drivers and examples for application developers.

2. How to use serial function

2.1. COM port Open(), Close()

Open and close the serial communication port to enable DTP7H-D Key,LED, Buzzer and switch operation.

```
m_comm=  new  CMycomm(_T("WWWW.WW")+m_str_comport,m_str_baudrate,_T("None"),_T("8  Bit"),_T("1
Bit"));
if( m_comm->Create(GetSafeHwnd()) != 0 ) {
    comport_state=true;
} else {
  AfxMessageBox(_T("COM PORT OPEN ERROR!"));
}
```

1.Parameter

Port

Serial port names to be used as a Serial Daemon.

Baudrate

The baudrate of the serial port.

Parity

The parity of the serial port.

Databit

The databit of the serial port.

Stopbit

The stopbit of the serial port.

2. Return value

If serial port open success, it returns handle. If it can't open, it returns 0.

3. Remark

DTP7H-D additional functions are controlled via serial. Through this manual, you can easily understand Daincube's sample application.

4. Requirements

Function	Required header	Refer to source code
CMycomm() Create()	Mycomm.h	Mycomm.cpp

5. Example

```
void CserialDlg::OnBnClickedBtConnect()
{
    if(comport_state) {    // Close COM port
    if(m_comm) {
      m_comm->Close();
      m_comm = NULL;
      comport_state=false;
  } else {      // Initial COM port
    m_comm= new CMycomm(_T("WWWW.WW")+m_str_comport,m_str_baudrate,_T("None"),_T("8
Bit"),_T("1 Bit"));
    if( m_comm->Create(GetSafeHwnd()) != 0 ) {
```

```
        comport_state=true;
      } else {
        AfxMessageBox(_T("COM PORT OPEN ERROR!"));
      }
    }
}
```

2.2.Send()

Send the serial packet for control to LED, Buzzer of DTP7H-D.

```
BOOL CMycomm::Send(char *outbuf, DWORD *len);
```

1.Parameter

outbuf

The buffer of serial packet to send.

len

The buffer length of serial packet to send.

2. Return value

If serial transmit success, it returns 1. If it can't fail, it returns 0.

3. Remark

LED or Buzzer is controlled by transmit the serial packet.

4. Requirements

Function	Required header	Refer to source code
Send()	Mycomm.h	Mycomm.cpp

5. Example

```
void CserialDlg::OnBnClickedBtLed1()
{
  // TODO: Add your control notification handler code here
  char buf_printf[10] = {0, };
  unsigned int crc_buf;
  DWORD dwBytes = 0;

  buf_printf[0] = STX;          // STX
  buf_printf[1] = MOD_SET;      // MOD (get : 0x10, set : 0x11)
  buf_printf[2] = SEL_LED;      // SEL (LED : 0x3A)
  buf_printf[3] = LEFT_LED1;      // Data1
  buf_printf[4] = LED_BLUE; // Data2 (off : 0x30, blue : 0x31, red : 0x32, all : 0x33)
  buf_printf[5] = DATA_RESERVED;      // Data3 (Reserved : 0x20)
  crc_buf = crc16_append(buf_printf,6);
  buf_printf[6] = (char)(crc_buf>>8)&0xff;
  buf_printf[7] = (char)crc_buf&0xff;
  buf_printf[8] = ETX;          // ETX
  buf_printf[9] = '₩0';

  dwBytes = strlen(buf_printf);
  m_comm->Send(buf_printf, &dwBytes);
}
```

2.3. Receive()

Receive the serial packet for control to Key, Switch of DTP7H-D.

```
int CMycomm::Receive(LPSTR inbuf, int len);
```

1.Parameter

inbuf

The buffer of serial packet to receive.

len

The buffer length of serial packet to receive.

2. Return value

If serial receive success, it return 1. If it can't fail, it returns 0.

3. Remark

Key is controlled by receive the serial packet.

4. Requirements

Function	Required header	Refer to source code
Receive()	Mycomm.h	Mycomm.cpp

5. Example

```
LRESULT CserialDlg::OnReceive(WPARAM length, LPARAM lpara)
{
  if(m_comm && comport_state) {

    while(length--)
    {
      m_comm->Receive(&g_Receive_Buffer[g_Head_Pointer],1);

      if(g_Head_Pointer >= BUFF_MAX-1)
        g_Head_Pointer = 0;
      else
        g_Head_Pointer++;
    }
  }

  return 0;
}
```

2.4. Keyboard Event receive()

Receive the keyboard event for control to LED, Buzzer of DTP7H-D.

```
void CserialDlg::OnRawInput(UINT nInputcode, HRAWINPUT hRawInput)
```

1.Parameter

nInputcode

A variable that checks whether an application has occurred while typing keyboard.

hRawInput

It is a structure that contains a device of Rawinput to process.

2. Return value

None.

3. Remark

Receives the generated keyboard event and controls LED or Buzzer.

4. Requirements

Function	Required header	Refer to source code
OnRawInput()	afxwin.h	serialDlg.cpp

5. Example

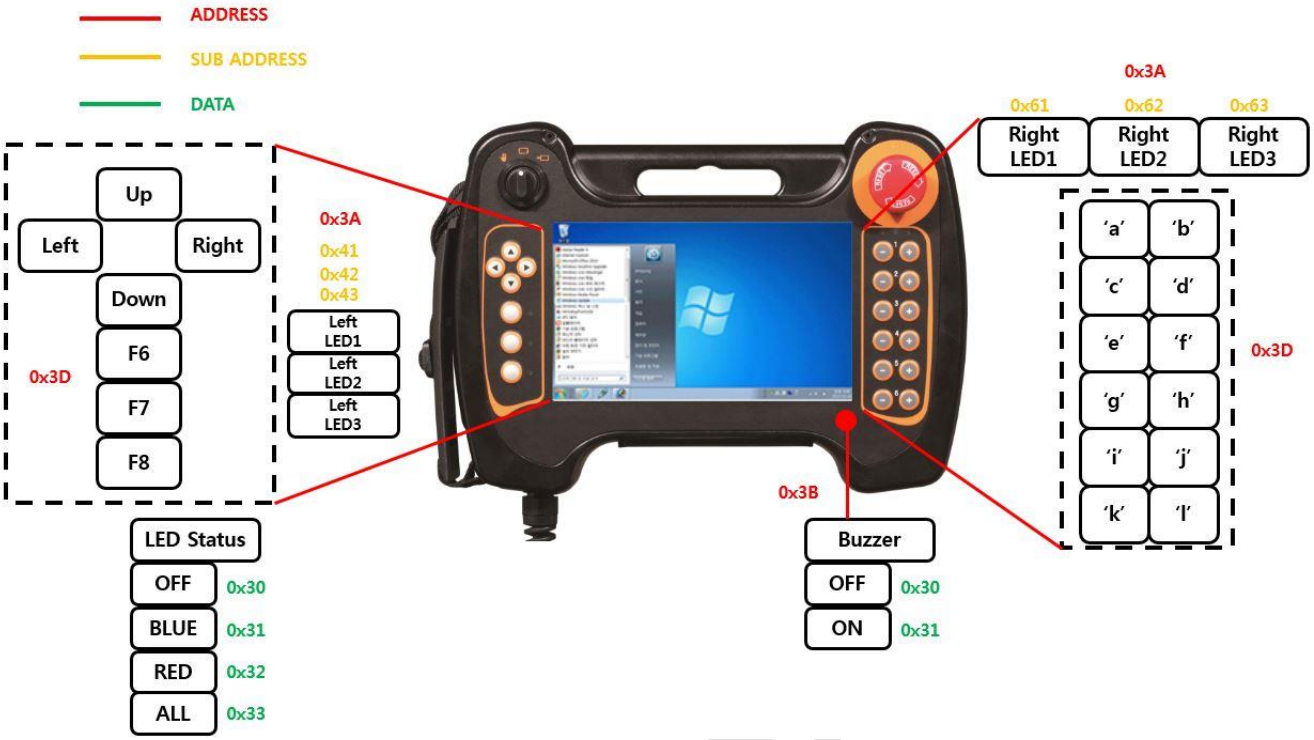
```
void CserialDlg::OnRawInput(UINT nInputcode, HRAWINPUT hRawInput)
{
  RAWINPUT input;
  char sel=0, data=0;
  memset(&input,0,sizeof(input));
  UINT sizeff=sizeof(RAWINPUT);
  GetRawInputData(hRawInput,RID_INPUT,&input,&sizeff,sizeof(RAWINPUTHEADER));

  if(input.header.dwType==RIM_TYPEKEYBOARD) {
    if(input.data.keyboard.Flags == 0) {      // Keyboard Down
      if ( (input.data.keyboard.VKey == 0xC1) ||
        (input.data.keyboard.VKey == 0xC4) ||
        (input.data.keyboard.VKey == 0xC7) ||
        (input.data.keyboard.VKey == 0xCA) ||
        (input.data.keyboard.VKey == 0xCD) ||
        (input.data.keyboard.VKey == 0xD0) ) {  // blue LED
        data = 0x1;
      }
            ····· omission
      else if (input.data.keyboard.VKey == 0xD3) {
        BUZ_Set(BUZZ_ON);
        return;
      }
    } else if (input.data.keyboard.Flags == 1) {  // Keyboard Up
      data = 0;
      if (input.data.keyboard.VKey == 0xD3) {
        BUZ_Set(BUZZ_OFF);
        return;
      }
    }
```

```
    // Virtual keycode 0xC1 ~ 0xD7 : Reserved
    switch(input.data.keyboard.VKey) {
      case 0xC1:    // blue
      case 0xC2:    // red
      case 0xC3:    // all
        sel = LEFT_LED1;
        break;
                ····· omission
      default:
        sel = -1;
        data = -1;
        break;
    }
    if( (sel>=0) && (data>=0) ) {
      LED_Set(sel, data);
    }
  }
  CDialog::OnRawInput(nInputcode, hRawInput);
}
```

3. ETC Driver information

3.1. ETC Driver information (Serial communication type)



3.2. ETC Driver information (Keyboard event type)

4. ETC Driver Control method

4.1. Method of LED control (Serial communication)

Control the LED of DTP7H-D by serial communication using Serial Daemon application provided by Daincube.

- Serial COM Port Open
- Create Packet Buffer
- Create CRC and save Packet Buffer
- Transmit Serial Packet

- Serial COM Port Close

By sending serial packet as below, LED of DTP7H-D can be controlled.

STX	MOD	SEL	DATA1	DATA2	DATA3	CRC_H	CRC_L	ETX
0x02	0x11	0x3A	0x41	0x33	0x20	0xXX	0xXX	0x03
1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE

MOD : 0x10 = MOD_GET, 0x11 = MOD_SET
SEL : 0x3A = SEL_LED
DATA1 : 0x41 = LEFT_LED1, 0x42 = LEFT_LED2, 0x43 = LEFT_LED3,
0x61 = RIGHT_LED1, 0x62 = RIGHT_LED2, 0x63 = RIGHT_LED3
DATA2 : 0x30 = OFF, 0x31 = BLUE, 0x32 = RED, 0x33 = ALL
DATA3 : 0x20 = Reserved

```
void CserialDlg::OnBnClickedBtLed1()
{
  // TODO: Add your control notification handler code here
  char buf_printf[10] = {0, };
  unsigned int crc_buf;
  DWORD dwBytes = 0;
  static char i = 0;

  buf_printf[0] = STX;          // STX
  buf_printf[1] = MOD_SET;        // MOD (get : 0x10, set : 0x11)
  buf_printf[2] = SEL_LED;        // SEL (LED : 0x3A)
  buf_printf[3] = LEFT_LED1;      // Data1
  if ( i == 3 )
    buf_printf[4] = LED_OFF;      // Data2 (off : 0x30, blue : 0x31, red : 0x32, all : 0x33)
  else
    buf_printf[4] = LED_BLUE + i;
  buf_printf[5] = DATA_RESERVED;    // Data3 (Reserved : 0x20)
  crc_buf = crc16_append(buf_printf,6);
  buf_printf[6] = (char)(crc_buf>>8)&0xff;
  buf_printf[7] = (char)crc_buf&0xff;
  buf_printf[8] = ETX;          // ETX
  buf_printf[9] = '₩0';

  dwBytes = strlen(buf_printf);
  m_comm->Send(buf_printf, &dwBytes);
}
```

See also

m_comm->Send : Reference "5.2 Send()"

4.2. Method of Buzzer control (Serial communication)

Control the buzzer of DTP7H-D by using Serial Daemon application provided by Daincube.

- Serial COM Port Open
- Create Packet Buffer
- Create CRC and save Packet Buffer
- Transmit Serial Packet
- Serial COM Port Close

By sending serial packet as below, Buzzer of DTP7H-D can be controlled.

STX	MOD	SEL	DATA1	DATA2	DATA3	CRC_H	CRC_L	ETX

0x02	0x11	0x3B	0x31	0x20	0x20	0xXX	0xXX	0x03
1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE

MOD : 0x10 = MOD_GET, 0x11 = MOD_SET
SEL : 0x3B = SEL_BUZZ
DATA1 : 0x30 = OFF, 0x31 = ON
DATA2 : 0x20 = Reserved
DATA3 : 0x20 = Reserved

```
void CserialDlg::OnBnClickedBtBuzzer()
{
  // TODO: Add your control notification handler code here
  char buf_printf[10] = {0, };
  unsigned int crc_buf;
  DWORD dwBytes = 0;
  static char i = 0;

  buf_printf[0] = STX;        // STX
  buf_printf[1] = MOD_SET;      // MOD (get : 0x10, set : 0x11)
  buf_printf[2] = SEL_BUZZ;     // SEL (BUZZ : 0x3B)
  if ( i == 0 )
    buf_printf[3] = BUZZ_ON;    // Data1 (off : 0x30, on : 0x31)
  else
    buf_printf[3] = BUZZ_OFF;
  buf_printf[4] = DATA_RESERVED;    // Data2 (Reserved : 0x20)
  buf_printf[5] = DATA_RESERVED;    // Data3 (Reserved : 0x20)
  crc_buf = crc16_append(buf_printf,6);
  buf_printf[6] = (char)(crc_buf>>8)&0xff;
  buf_printf[7] = (char)crc_buf&0xff;
  buf_printf[8] = ETX;        // ETX
  buf_printf[9] = 'WO';

  dwBytes = strlen(buf_printf);
  m_comm->Send(buf_printf, &dwBytes);
}
```

See also

m_comm->Send : Reference "5.2 Send()"

4.3. Method of Keypad control (Serial communication)

Receive DTP7H-D keypad event using Serial Daemon Application provided by Daincube.
- Serial COM Port Open
- Create Packet Receive Buffer
- Receive Serial Packet
- Check to serial Packet and parsing Data
- Serial COM Port Close

By receiving serial packet as below, you can check DTP7H-D keypad and switch status.

STX	MOD	SEL	DATA1	DATA2	DATA3	CRC_H	CRC_L	ETX

0x02	0x10	0x3C	0x30	0x33	0x30	0xXX	0xXX	0x03
1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE

MOD : 0x10 = MOD_GET
SEL : 0x3D = SEL_KEYPAD
DATA1 : 0x30 = KEYPAD_UP, 0x31 = KEYPAD_DOWN
DATA2 : KEY_A = 30, KEY_B = 48, KEY_C = 46, KEY_D = 32, KEY_E = 18, KEY_F = 33, KEY_G = 34, KEY_H = 35, KEY_I = 23, KEY_J = 36, KEY_K = 37, KEY_L = 38, KEY_DOWN = 108, KEY_UP = 103, KEY_RIGHT = 106, KEY_LEFT = 105, KEY_F6 = 64, KEY_F7 = 65, KEY_F8 = 66, KEY_F9 = 67
DATA3 : 0x20 = DATA_RESERVED

```
UINT CserialDlg::OperThread(LPVOID aParam)
{
  CserialDlg *dlg = (CserialDlg*)aParam;
  unsigned int crc_buf;
  DWORD keyevent_buf;

  while(dlg->g_Is_Thread_Run)
  {

    ····· omission

    if ( ((dlg->g_Packet_Buffer[0] != STX) || (dlg->g_Packet_Buffer[8] != ETX)) ){    //STX, ETX
Check
      continue;
    }

    if(dlg->g_Packet_Buffer[1] != MOD_GET){   //MOD Check
      continue;
    }

    if ( dlg->g_Packet_Buffer[2] != SEL_KEYPAD ){   // SEL (KEY : 0x3D)
    ····· omission
      continue;
    }

    crc_buf = dlg->crc16_append(dlg->g_Packet_Buffer,6);

    if((dlg->g_Packet_Buffer[6]!=(char)((crc_buf>>8)&0xff)) || (dlg-
>g_Packet_Buffer[7]!=(char)(crc_buf&0xff))){  //CRC Check
      continue;
    }

    if ( dlg->g_Packet_Buffer[3] == KEYPAD_DOWN ) {     // Key DOWN
      keyevent_buf = 0;
    }
    else if ( dlg->g_Packet_Buffer[3] == KEYPAD_UP ) {    // Key UP
      keyevent_buf = KEYEVENTF_KEYUP;
    }

    switch ( dlg->g_Packet_Buffer[4] ) {
      case KEY_A :  keybd_event(0x41,0,keyevent_buf,0); break;
      case KEY_B :  keybd_event(0x42,0,keyevent_buf,0); break;
      case KEY_C :  keybd_event(0x43,0,keyevent_buf,0); break;
      case KEY_D :  keybd_event(0x44,0,keyevent_buf,0); break;
      case KEY_E :  keybd_event(0x45,0,keyevent_buf,0); break;
```

```
      case KEY_F :  keybd_event(0x46,0,keyevent_buf,0); break;
      case KEY_G :  keybd_event(0x47,0,keyevent_buf,0); break;
      case KEY_H :  keybd_event(0x48,0,keyevent_buf,0); break;
      case KEY_I :  keybd_event(0x49,0,keyevent_buf,0); break;
      case KEY_J :  keybd_event(0x4A,0,keyevent_buf,0); break;
      case KEY_K :  keybd_event(0x4B,0,keyevent_buf,0); break;
      case KEY_L :  keybd_event(0x4C,0,keyevent_buf,0); break;
      case KEY_UP : keybd_event(0x26,0,keyevent_buf,0); break;
      case KEY_DOWN:  keybd_event(0x28,0,keyevent_buf,0); break;
      case KEY_LEFT:  keybd_event(0x25,0,keyevent_buf,0); break;
      case KEY_RIGHT: keybd_event(0x27,0,keyevent_buf,0); break;
      case KEY_F6 : keybd_event(0x75,0,keyevent_buf,0); break;
      case KEY_F7 : keybd_event(0x76,0,keyevent_buf,0); break;
      case KEY_F8 : keybd_event(0x77,0,keyevent_buf,0); break;
      case KEY_F9 : keybd_event(0x78,0,keyevent_buf,0); break;
    }

  }

  return 0;
}
```

See also

m_comm->Receive : **Reference "5.3 Receive()"**

4.4. Method of LED, Buzzer control (Keyboard event receive)

Control the DTP7H-D LED and Buzzer with keyboard events using Serial Daemon Application provided by Daincube.
You must implement a program that generates a keyboard event. This manual explains how the Serial Daemon Application works in relation to keyboard event reception.

- Serial COM Port Open
- Keyboard event operate to Virtual key mapping
- Receive Keyboard event
- Operate LED, Buzzer

DTP7H-D LED can be controlled by receiving keyboard event as bellow.

```
void CserialDlg::OnRawInput(UINT nInputcode, HRAWINPUT hRawInput)
{
  RAWINPUT input;
  char sel=0, data=0;
  memset(&input,0,sizeof(input));
  UINT sizeff=sizeof(RAWINPUT);
  GetRawInputData(hRawInput,RID_INPUT,&input,&sizeff,sizeof(RAWINPUTHEADER));

  if(input.header.dwType==RIM_TYPEKEYBOARD) {
    if(input.data.keyboard.Flags == 0) {      // Keyboard Down
      if ( (input.data.keyboard.VKey == 0xC1) ||
        (input.data.keyboard.VKey == 0xC4) ||
        (input.data.keyboard.VKey == 0xC7) ||
        (input.data.keyboard.VKey == 0xCA) ||
        (input.data.keyboard.VKey == 0xCD) ||
        (input.data.keyboard.VKey == 0xD0) ) {  // blue LED
        data = 0x1;
      }
            ····· omission
      else if (input.data.keyboard.VKey == 0xD3) {
        BUZ_Set(BUZZ_ON);
        return;
      }
    } else if (input.data.keyboard.Flags == 1) {  // Keyboard Up
      data = 0;
      if (input.data.keyboard.VKey == 0xD3) {
        BUZ_Set(BUZZ_OFF);
        return;
      }
    }

    // Virtual keycode 0xC1 ~ 0xD7 : Reserved
    switch(input.data.keyboard.VKey) {
      case 0xC1:    // blue
      case 0xC2:    // red
      case 0xC3:    // all
        sel = LEFT_LED1;
        break;
              ····· omission
      default:
        sel = -1;
        data = -1;
        break;
    }
    if( (sel>=0) && (data>=0) ) {
      LED_Set(sel, data);
    }
  }
  CDialog::OnRawInput(nInputcode, hRawInput);
}
```


See also

OnRawInput : Reference "**5.4 Keyboard Event Receive()**"

4.5. Method of Touch, Keypad (Enable/ Disable) control (Serial communication)

Use the Serial Daemon program provided by Daincube to control Enable / Disable of DTP7H-D Touch and keypad..

- Serial COM Port Open
- Create Packet Buffer
- Create CRC and save Packet Buffer
- Transmit Serial Packet
- Serial COM Port Close

Enable / Disable of DTP10-D touch and keypad can be controlled by sending / receiving serial packet as below.

STX	MOD	SEL	DATA1	DATA2	DATA3	CRC_H	CRC_L	ETX
0x02	0x11	0x3E	0x81	0x01	0x30	0xXX	0xXX	0x03
1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE	1BYTE

MOD : MOD_SET = 0x11

SEL : SEL_KEYPAD = 0x3E

DATA1 : KEYPAD_MODE = 0x82, TOUCH_MODE = 0x81

DATA2 : MODE_ENABLE = 0x01, MODE_DISABLE = 0x02

DATA3 : DATA_RESERVED = 0x20

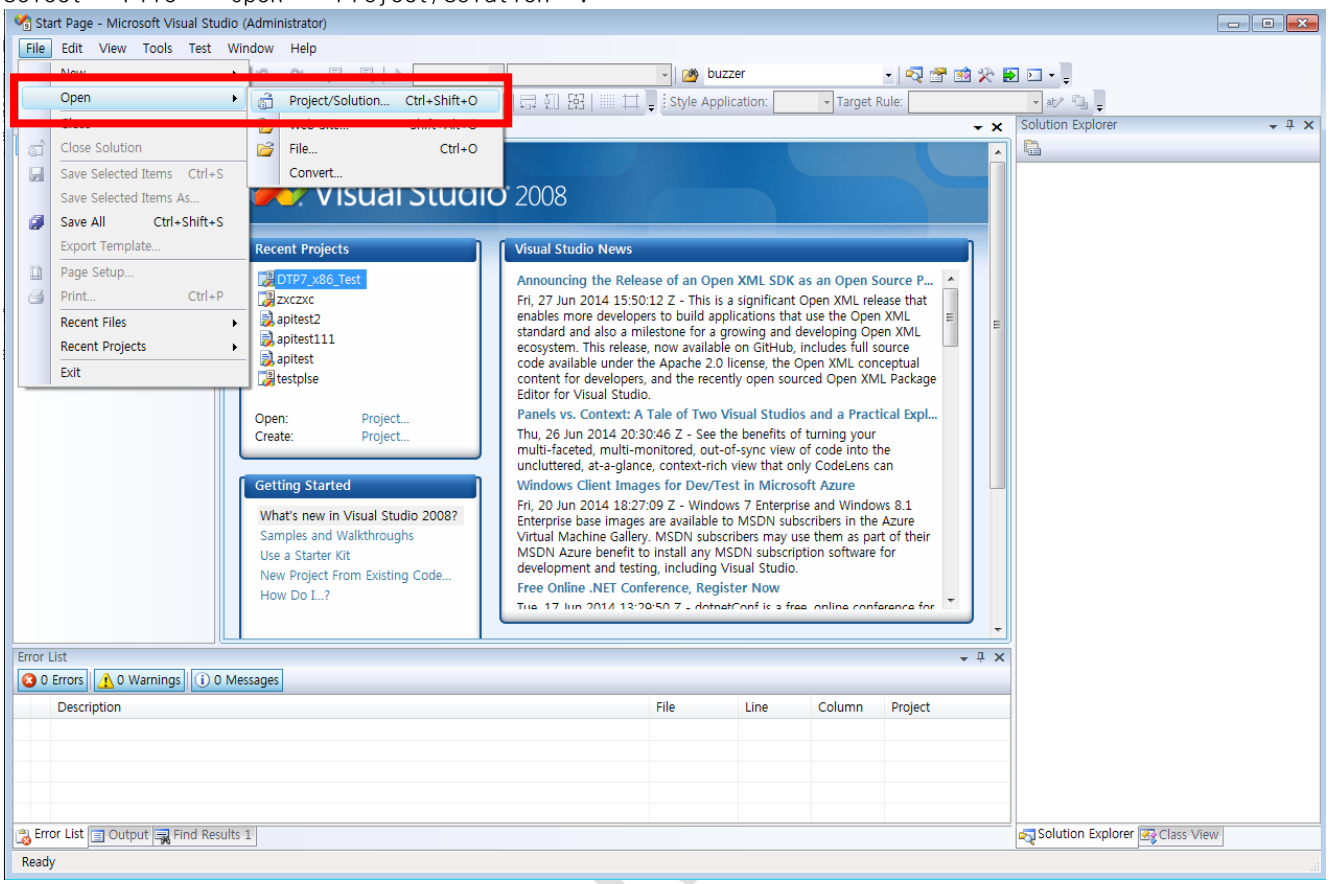
```
void CserialDlg::OnBnClickedBtEnable()
{
   // TODO: Add your control notification handler code here
   char buf_printf[10] = {0, };
   unsigned int crc_buf;
   DWORD dwBytes = 0;
   if(((CButton*)GetDlgItem(IDC_CHECK1_Touch))->GetCheck())
   {
      buf_printf[0] = STX;          // STX
      buf_printf[1] = MOD_SET;       // MOD (get : 0x10, set : 0x11)
      buf_printf[2] = SEL_TOUCH_KEY;       // SEL (TOUCH,KEY : 0x3E)
      buf_printf[3] = TOUCH_MODE;
      buf_printf[4] = MODE_ENABLE;      // Data2 (Reserved : 0x20)
      buf_printf[5] = DATA_RESERVED;    // Data3 (Reserved : 0x20)
      crc_buf = crc16_append(buf_printf,6);
      buf_printf[6] = (char)(crc_buf>>8)&0xff;
      buf_printf[7] = (char)crc_buf&0xff;
      buf_printf[8] = ETX;          // ETX
      buf_printf[9] = '₩0';
      dwBytes = strlen(buf_printf);
      m_comm->Send(buf_printf, &dwBytes);
   }

   if(((CButton*)GetDlgItem(IDC_CHECK2_Keypad))->GetCheck())
   {
      buf_printf[0] = STX;          // STX
      buf_printf[1] = MOD_SET;       // MOD (get : 0x10, set : 0x11)
      buf_printf[2] = SEL_TOUCH_KEY;       // SEL (TOUCH,KEY : 0x3E)
      buf_printf[3] = KEYPAD_MODE;
      buf_printf[4] = MODE_ENABLE;      // Data2 (Reserved : 0x20)
      buf_printf[5] = DATA_RESERVED;    // Data3 (Reserved : 0x20)
      crc_buf = crc16_append(buf_printf,6);
      buf_printf[6] = (char)(crc_buf>>8)&0xff;
      buf_printf[7] = (char)crc_buf&0xff;
      buf_printf[8] = ETX;          // ETX
      buf_printf[9] = '₩0';
      dwBytes = strlen(buf_printf);
      m_comm->Send(buf_printf, &dwBytes);
   }
}
```

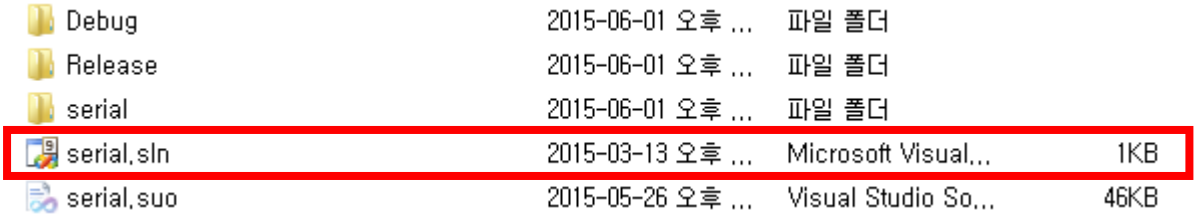
5. Method of Serial Daemon project build

5.1. Project open

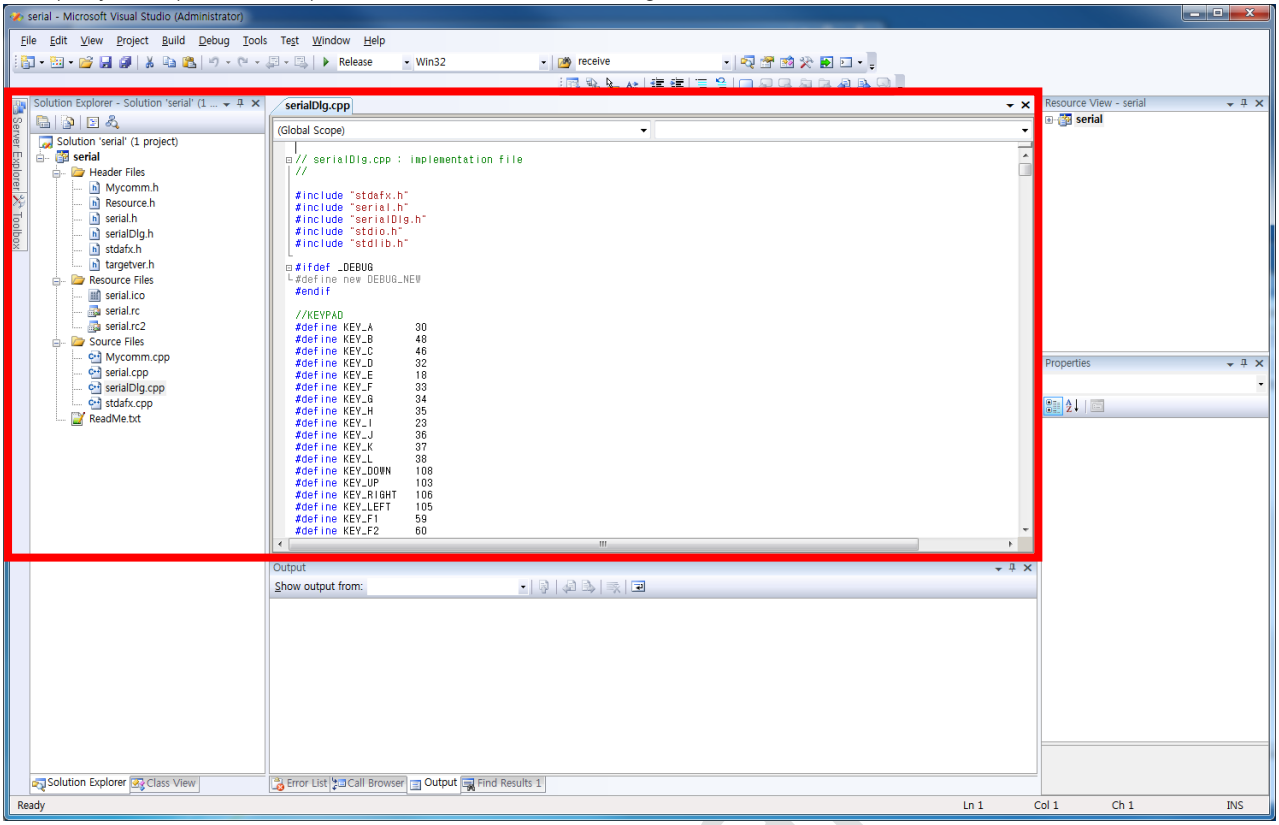
Select "File -> Open -> Project/Solution".



Choice "02_DTP7H-P_SW or 02_DTP7H-D_SW >> 03_Example >> 01_DTP7H-PD_SerialDaemon >> serial.sln" solution file in SDK CD.



The project open to complete. Check the following screen.



5.2. Project build

Click the "Build -> Build Solution" button. Check the following screen.